**CV Practical No.: 3A**

**Aim: Implementation of low pass filters on a colored image**

**New Concept:**

**i. kernel\_size:** it refers to the dimensions of a kernel or filter. A kernel is a small matrix used to apply operations like blurring, sharpening, or edge detection to an image. It determines the neighborhood of pixels that will be considered when calculating the output pixel value. For example, a kernel\_size of (3, 3) means a 3x3 matrix is used, affecting a pixel and its 8 immediate neighbors.

**ii. apply\_average\_filter:** it is a process that replaces each pixel's value with the average of the pixel values within a specified neighbourhood. This filter smooths an image by reducing sharp transitions in pixel intensities, effectively blurring it.

**iii. cv2.blur:** it is a function from the OpenCV that implements an average filter. It takes an image and a kernel size as input and returns a blurred version of the image. It performs the same function as the apply\_average\_filter.

**iv. sigmaX:** it is a parameter used in Gaussian blurring, representing the standard deviation of the Gaussian kernel in the horizontal (X) direction. It controls the amount of blurring in the horizontal direction. A larger sigmaX results in more blurring.

**v. cv2.GaussianBlur:** it is a function from OpenCV that applies a Gaussian blur to an image. It uses a Gaussian kernel, which gives more weight to the central pixels and less weight to the surrounding pixels, resulting in a more natural-looking blur compared to a simple average blur.

**vi. apply\_weighted\_average\_filter:** it is a generalization of the average filter where the pixels within the kernel are assigned different weights before averaging. This allows for more control over the blurring effect. Gaussian blurring is an example of a weighted average filter.

**vii. median\_filtered\_image:** it is the result of applying a median filter to an image. The median filter replaces each pixel's value with the median value of the pixels within a specified neighborhood. It's particularly effective at removing salt-and-pepper noise.

**viii. cv2.medianBlur:** it is a function from OpenCV that applies a median filter to an image. It takes an image and a kernel size as input and returns the median-filtered image.

**Theory:**

**i. Average filter:**

* The average filter is a simple linear filter used for smoothing images. It operates by replacing each pixel's value with the average (mean) of the pixel values within a defined neighborhood around that pixel.
* This neighborhood is typically a rectangular or square window, defined by the kernel\_size.
* It essentially smooths out local variations in pixel values.
* All pixels within the kernel contribute equally to the output.

**ii. Gaussian blur:**

* Gaussian blur is a linear smoothing filter that uses a Gaussian function as its kernel.
* The Gaussian function gives more weight to pixels closer to the center of the kernel and less weight to pixels farther away.
* Gaussian blur is a linear smoothing filter that uses a Gaussian function as its kernel.
* The Gaussian function gives more weight to pixels closer to the center of the kernel and less weight to pixels farther away.

**iii. Median blur:**

* The median blur is a non-linear filter that replaces each pixel's value with the median value of the pixels within a defined neighborhood.
* Unlike the average or Gaussian filters, it doesn't use a weighted average; instead, it sorts the pixel values within the kernel and selects the middle value.
* It's a statistical filter that uses the median, which is a robust measure of central tendency.
* It's highly effective at removing outliers without significantly blurring edges.

**Program:**

**Program1: Low filter image:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

def apply\_average\_filter(image\_path, kernel\_size):

# Read the image

image = cv2.imread(image\_path)

# Apply the average filter

filtered\_image = cv2.blur(image, (kernel\_size, kernel\_size))

return filtered\_image

if \_\_name\_\_ == "\_\_main\_\_":

# Input image path and kernel size

input\_image\_path = "C:/Users/admin/Desktop/Nupur/scenery.jpg"

kernel\_size = 10 # You can adjust the kernel size as needed.

# Apply the average filter

filtered\_image = apply\_average\_filter(input\_image\_path, kernel\_size)

# Display the original and filtered images

#cv2.imshow("Original Image", cv2.imread(input\_image\_path))

#cv2.imshow("Filtered Image", filtered\_image)

#Image1

plt.figure(figsize = (10,5))

plt.subplot(1,2,2)

#plt.imshow(cv2.imread(input\_image\_path))

plt.imshow(cv2.imread(input\_image\_path), cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(1,2,1)

#plt.imshow(filtered\_image)

plt.imshow(filtered\_image, cmap = 'gray')

plt.title('Filtered Image')

plt.axis('off')

plt.show()

# Wait for a key press and then close the windows

cv2.waitKey(0)

cv2.destroyAllWindows()

**Program2: Gaussian filter:**

import cv2

import matplotlib.pyplot as plt

def apply\_weighted\_average\_filter(image\_path, kernel\_size, sigmaX):

# Read the image

image = cv2.imread(image\_path)

# Apply the weighted average filter (Gaussian blur)

filtered\_image = cv2.GaussianBlur(image, (kernel\_size, kernel\_size), sigmaX)

return filtered\_image

if \_\_name\_\_ == "\_\_main\_\_":

# Input image path, kernel size, and sigmaX (standard deviation in X direction)

input\_image\_path = "C:/Users/admin/Desktop/Nupur/scenery.jpg"

kernel\_size = 5 # You can adjust the kernel size as needed.

sigmaX = 4 # You can adjust the sigmaX value for the Gaussian kernel.

# Apply the weighted average filter

filtered\_image = apply\_weighted\_average\_filter(input\_image\_path, kernel\_size, sigmaX)

# Display the original and filtered images

#cv2.imshow("Original Image", cv2.imread(input\_image\_path))

#cv2.imshow("Filtered Image", filtered\_image)

#Image1

plt.figure(figsize = (10,5))

plt.subplot(1,2,2)

#plt.imshow(cv2.imread(input\_image\_path))

plt.imshow(cv2.imread(input\_image\_path), cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(1,2,1)

#plt.imshow(filtered\_image)

plt.imshow(filtered\_image, cmap = 'gray')

plt.title('Filtered Image')

plt.axis('off')

plt.show()

# Wait for a key press and then close the windows

cv2.waitKey(0)

cv2.destroyAllWindows()

**Program3: Median filter:**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Load the image

image\_path = "C:/Users/admin/Desktop/Nupur/scenery.jpg"

original\_image = cv2.imread(image\_path)

# Apply median filter with specified kernel size

kernel\_size = 3 # Adjust this value to change the filter size

median\_filtered\_image = cv2.medianBlur(original\_image, kernel\_size)

# Display the original and filtered images

#cv2.imshow('Original Image', original\_image)

#cv2.imshow('Median Filtered Image', median\_filtered\_image)

#Image1

plt.figure(figsize = (10,5))

plt.subplot(1,2,2)

plt.imshow(original\_image)

#plt.imshow(original\_image, cmap = 'gray')

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(1,2,1)

plt.imshow(median\_filtered\_image)

#plt.imshow(median\_filtered\_image, cmap = 'gray')

plt.title('Median Filtered Image')

plt.axis('off')

plt.show()

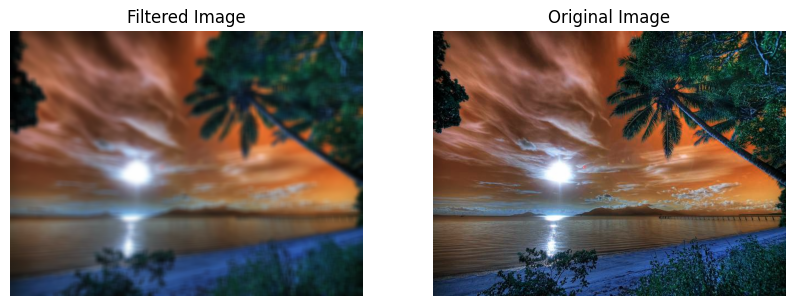
# Wait for a key press and close the windows

cv2.waitKey(0)

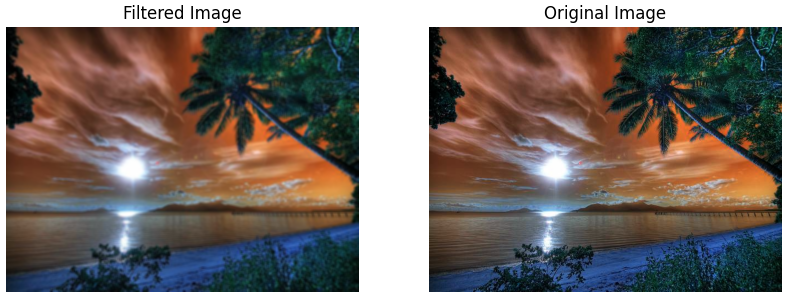
cv2.destroyAllWindows()

**Output:**

**Program1:**



**Program2:**



**Program3:**

